On-line Generation of Fractal and MultiFractal Traffic

Darryl Veitch[1], **Jon-Anders Bäckar**[2], **Jens Wall**[2] **Jennifer Yates**[3] and **Matthew Roughan**[1]

1 - Department of Electrical Engineering, University of Melbourne, Victoria 3010, Australia.
E-mail: {m.roughan,d.veitch}@ee.mu.oz.au

2 - Formally at SERC, RMIT University, Level 3, 110 Victoria St, Carlton, Vic 3053, Australia

3 - AT&T Research Laboratories, 180 Park Avenue, P.O. Box 971, Florham Park, NJ 07932-0000, USA
E-mail: jyates@research.att.com

Abstract— **A general wavelet based framework is described for the on-line generation of time-series, particularly fractal and certain multifractal time series. A scalable method is given to transmit a given time series as a cell stream over ATM at OC3 rates on inexpensive hardware - a low end PC. It is based on the CORAL system with `FreeBSD` drivers written for a standard ATM NIC. The on-line systems were then coupled to faithfully generate and transmit synthetic fractal and multifractal traffic at high rates with very low memory requirements. The system is highly scalable and could be the basis of an inexpensive test traffic generator.**

I. Introduction

Many studies of data traffic have confirmed its fractal or self-similar nature [3], [4], [21], and more recently its multifractal characteristics [7], [11], [27], [26]. Such characteristics may have important consequences for the performance of networks, and hence simplistic traffic models may produce misleading test results. However, most standard test equipment used to generate reference loads for benchmarking network elements is incapable of generating traffic with fractal characteristics. Instead traditional models are used which are very poor at describing the real fractal nature of traffic.

This paper presents a system for generating realistic aggregate test traffic, in particular for *Asynchronous Transfer Mode* (ATM) operating over 155 Mbps OC3 links. The method is highly adaptable, and allows generation of traffic with fractal or multifractal characteristics. Among other uses, such traffic could be employed to test QoS characteristics of network equipment in the presence of realistic traffic loads, rather than with the undemanding test traffic most test and measurement devices generate.

At least one group has produced self-similar test traffic using the heavy-tailed On/Off paradigm and variants, for example [31], [33], [12], [13]. We have implemented this method of traffic generation [32] and found it suffers from several defects, including its inability to generalise to multifractal behaviour. Instead, this paper will focus on using the inverse discrete wavelet transform to construct a fractal, or multi-fractal time series, as described by Riedi *et al* [10]. It has been noted by several authors (for example [35], [14]), that the wavelet transform is ideally suited for such a purpose. In this paper we exploit one of its key benefits, that it can be implemented on-line. The time-series is generated in real-time and passed to a simple ATM Network Interface Card (NIC) which uses purpose designed firmware to generate a stream of ATM cells which very closely match the traffic rate as specified by the time series. The advantage of this method is that a sequence of test traffic of any length, with detailed fractal properties, can be generated with scalable memory and processing requirements. The decoupling of the time-series generation and traffic generation phases is a key advantage. It means that the series could be generated by alternative mechanisms, and could even be generated off-line, or be derived from high-resolution traffic logs. On the other hand the time series generator output could be used to feed a simulation, or be stored to file. Finally the on-line wavelet inversion framework used is itself general and could be used to synthesize according to other models, including non-fractal models, on-line and in real-time.

The design of the traffic generation half of the system is based on that of the CORAL project [5], [6], with a `FreeBSD` transmission driver written for the first time based on existing `DOS` drivers, with some corrections and modifications. The aim is to reproduce as accurately as possible at the ATM cell level the bit rate specified by the time series, here specified at a resolution of one value per 1ms (this parameter can be changed). To within the accuracy of the traffic measurement system used, this aim was achieved. The approach is to generate IP packets which are then transmitted via AAL5 over ATM. In the current form the IP packets are generated at uniform intervals, with the traffic rate being determined by the packet size, though this rather simplified method of generation is not a requirement of the hardware, merely a first stage in the implementation. The hardware used in the generator is based around a low end Intel PC running `FreeBSD`, and a commodity off-the-shelf ATM NIC (from FORE Systems), the total cost being very low at around $4000 (Australian). The test generator can generate ATM traffic in real-time at close to the line rate, making it a cheap, effective method of generating traffic.

The current performance of the system indicates that it would be possible to place multiple NICs in the PC to allow generation of several simultaneous traffic sources from the same box. This could lead to its use both as an effective test mechanism for real networks as well as in benchmarking isolated systems.

In section 2 we address the on-line generation of the time series, including the necessary wavelet background. In section 3 it is shown how a time series can be sent as an ATM cell stream in real-time, by using purpose built firmware. In section 4 these separate halves are combined into a working system and an example is given of a simple multifractal model which falls naturally within the framework. Scalability issues are also discussed, before concluding in section 5.

[123]The majority of this work was conducted while the authors were at the Software Engineering Research Centre at RMIT University, Melbourne. The work was sponsored by Ericsson Australia.

II. Generating the Time Series

A. Long Range Dependence and Multifractals

Models for traffic are often defined within the following framework. Consider a stationary time series (discrete time stochastic process) $X(k)$, $k = 1, 2\cdots$, representing for example the number of bytes per time interval observed on a given link under steady traffic conditions. An important fractal property found in such series is that of *long-range dependence*, which can be defined as the slow, power-law like decrease at large lag of the autocovariance function, or equivalently as the power-law divergence at the origin of its spectrum:

$$f_x(\nu) \sim c_f|\nu|^{-\alpha}, \quad |\nu| \to 0. \tag{1}$$

This long memory property represents a replacement of the paradigm of a time constant or characteristic time scale by the notion of scale invariant, or 'constant' relationships *between* scales, controlled by the dimensionless *scaling exponent* α. This description is closely associated to *self-similarity* and the associated exponent, the *Hurst parameter* (see [8], [20] for more details of these concepts and their inter-relationships). Long-range dependence (LRD) however, being a second order characterisation, is not a complete description of the statistics of $x(k)$, unless the series is Gaussian. This is frequently not the case, and it has been observed [7], [27], [11] that richer scaling behaviour known as *multifractal scaling* arises in wide area network traffic, which implies non-trivial scaling behaviour not just in the second order statistics, but in statistics of all orders. This implies in turn that the single exponent α must be replaced by a selection, indeed a spectrum, of exponents. It is beyond the scope of this paper to describe multifractal processes in detail (see [9], [10], [22] for more details), however we briefly describe a class of objects known as *conservation binomial cascades*, which constitute an important subclass of multifractals. A (random) binomial cascade is an iterative re-distribution of an originally uniform mass on the interval, where line segments are repeatedly divided in two, the masses being re-distributed via weights obtained by multiplying the original weights by *multipliers* resulting from independent trials of a given random variable. If the weights are such that the mass is conserved on average, then the cascade is *conservative.* Continuing this procedure recursively for a number of levels generates a highly irregular, non-Gaussian signal with scaling properties which is a multifractal in the limit. Note that although a spectrum of exponents is now involved, these can be expressed in specific examples as parametric functions of a small number of 'model' parameters, preserving the need for parsimony. A wavelet implementation of a simple random binomial cascade due to Riedi *et al* [10] will be described below.

B. Multiresolution Analysis and Wavelets

Wavelets are analysing functions which are localised in both time and *scale* or frequency. Wavelets have become a large subject in their own right and are used in diverse ways. For our purposes only a subset of wavelets will be considered, those falling within the *Multiresolution Analysis* (MRA) theory [25] leading to the *Discrete Wavelet Transform*, where the primary object is not the *mother wavelet*, $\psi_0(t)$, but the *scaling function* $\phi_0(t)$, $t \in \mathcal{R}$.

The collection of integer translates of the scaling function, $\{\phi_0(t-k), k \in \mathcal{Z}\}$ span an *approximation* subspace V_0 of square integrable functions $L^2(\mathcal{R})$. The special, defining property of $\phi_0(t)$ is that for any real function or sample path of a stochastic process $x(t)$ lying in V_0, the 'coarser' dilated function $x(t/2)$ is also in V_0, in a subset called V_1. This leads to the recursive definition of a set of nested subspaces V_j such that $V_j \subset V_{j-1}$, whose intersection is the null set in the limit $j = \infty$, and union the full space $L^2(\mathcal{R})$ at $j = -\infty$.

The multiresolution analysis of x involves successively projecting it into each of the approximation subspaces V_j: $\text{approx}_j(t) = (\text{Proj}_{V_j} x)(t) = \sum_k a_x(j,k)\phi_{j,k}(t)$, where the $\phi_{j,k}$, defined by $\{\phi_{j,k}(t) = 2^{-j/2}\phi_0(2^{-j}t - k),\ k \in \mathcal{Z}\}$ are scaled and translated versions of the scaling function which act as basis functions, and the $a_x(j,k)$ are the corresponding *approximation coefficients.* It is not necessary to perform all of these projections directly however. The information removed when going from one approximation to the next, coarser one, is called the detail: $\text{detail}_j(t) = \text{approx}_{j-1}(t) - \text{approx}_j(t)$. The MRA shows that the detail signals $\text{detail}_j(t)$ belong to the complementary subspaces $W_j = V_j \ominus V_{j-1}$, called the wavelet subspaces, for which the *mother wavelet* ψ_0 plays the role of canonical basis function, analogous to that of the scaling function for the V_j. It is therefore possible to decompose any given approximation into a detail and a new, coarser, approximation. The end result is a recursive decomposition of the initial approximation, which we call approx_0 by convention, into a set of details of decreasing resolution, expanded in wavelet functions with corresponding wavelet *detail* coefficients $d_x(j,k)$, and a final, most coarse approximation, expanded in scaling functions:

$$\begin{aligned} \text{approx}_0(t) &= \text{approx}_J(t) + \textstyle\sum_{j=1}^J \text{detail}_j(t) \\ &= \textstyle\sum_k a_x(J,k)\phi_{J,k}(t) \\ &+ \textstyle\sum_{j=1}^J \sum_k d_x(j,k)\psi_{j,k}(t). \end{aligned} \tag{2}$$

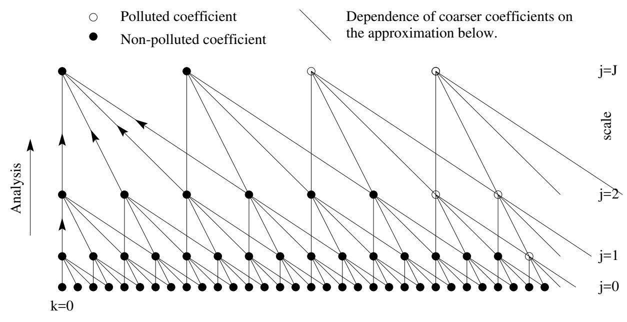
Varying J simply means deciding if more or less information contained in $\text{approx}_0(t)$ is written in details as opposed to the final approximation approx_J, and involves no loss of information. Information is unavoidably lost however in the initialising projection of $x(t)$ into V_0. Indeed in practice one generally deals with discretized data $x(k)$ and the initialisation is approximated at the finest resolution available by setting $a_0(k) = x(k)$ (see however [17]).

If we expand dilated versions of ψ_0 and ϕ_0 themselves in terms of ϕ_0 then we obtain the so called two-scale equations:

$$\begin{aligned} \phi(t/2) &= \sqrt{2}\textstyle\sum_n u_n\phi(t-n) \\ \psi(t/2) &= \sqrt{2}\textstyle\sum_n v_n\phi(t-n) \end{aligned} \tag{3}$$

from which the following relationships relating the approximation and detail coefficients at adjacent levels follow easily:

$$\begin{aligned} a_{j,k} &= \textstyle\sum_n u_n a_{j-1,2k+n} \\ d_{j,k} &= \textstyle\sum_n v_n a_{j-1,2k+n}. \end{aligned} \tag{4}$$



○ Polluted coefficient
● Non-polluted coefficient
Dependence of coarser coefficients on the approximation below.
Analysis
scale
j=J
j=2
j=1
j=0
k=0

Fig. 1. **Coefficient relationships, Analysis.** Derivation of the approximation **and** detail coefficients at the upper level $j+1$, from the approximation coefficients at the finer level j, for generating sequences $u(k)$, $v(k)$ of length four. Unfilled circles indicate *polluted coefficients*, those calculated in part from missing data.

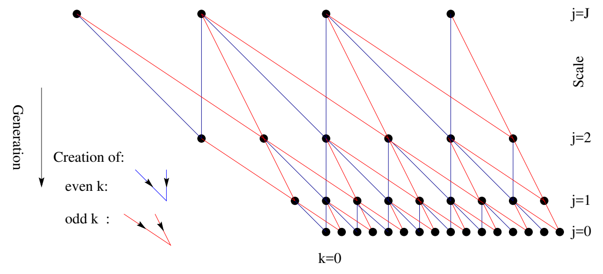


Fig. 2. **Coefficient relationships, Generation.** Lines showing dependence of the approximation coefficients **only** at the lower (finer) level j, on the approximation and detail coefficients at level $j+1$, for generating sequences $u(k)$, $v(k)$ of length four.

Two essential facts now emerge. First, we do not need to deal with the time shapes of wavelets or scaling functions, but only with the discrete filters u_n and v_n, which are said to generate the multiresolution, and control all its properties. In fact in wavelet design the sequence u_n is dealt with directly. Because of the highly localized nature of $\phi(t)$ and $\psi(t)$, most of these sequence elements are negligible, and in some cases, including all the examples in this paper, the scaling function and mother wavelet have finite time support, so only a finite number are non-zero. Second, from the initial approximation sequence $a_0(k)$, we can deduce **all** of the other approximation and detail coefficients via simple discrete operations.

The recursions given by (4) define the approximation and detail coefficients at a coarser level $j+1$, through the approximation coefficients at level j from which they came, as illustrated in figure 1 in a case where the sequences $u(k)$ and $v(k)$ are zero except for $k = 0$, 1, 2 and 3. In this figure, and in figures 2 and 3, the circles are on the *dyadic grid*, a set of points given by $\{(t, \text{scale}) = (k2^j, 2^j),\ j, k \in \mathcal{Z}\}$ in the *time-scale plane*, corresponding to the locations around which the $\psi_{j,k}$ and $\phi_{j,k}$ are centred and the details and approximation coefficients belong. Each row corresponds to a fixed scale 2^j, beginning with $j = 0$ on the bottom, within which points are indexed by k. The recursion relations above are in the direction required for *analysis*, when one begins with a function $x(t)$, initializes to obtain $a_0(k)$, and then proceeds 'upward' to calculate all the detail coefficients. The analysis then consists of studing their statistical properties, for example the measurement of the exponent α can be made by considering the logarithm of the variance of $d_{j,\cdot}$ as a function of j [20], [24]. To invert this procedure, that is to begin from the approximation and detail coefficients at the coarsest level and to *reconstruct* the finer level approximations, we need the inverse relation

$$a_{j,k} = \sum_n u_{k-2n} a_{j+1,n} + v_{k-2n} d_{j+1,n} \qquad (5)$$

illustrated in figure 2, again for finite generating sequences or *filters* of length four. This last relation assumes that the details are already available, which would be the case if a prior analysis phase had calculated and stored them. Our interest however is in series *generation*, so the details must somehow be generated at each level, as well as a J value selected, and a most coarse sequence $a_J(k) = a_{J,k}$ supplied.

It is important to clarify the issue of *edge effects*. If $x(t)$ were known for all time, then so would the detail coefficients $d_{j,k}$ for each $\{j, k \in \mathcal{Z}\}$, and J could be arbitrarily large. Since in practice the length n of $a_0(k)$ is finite, and the density of coefficients halves with each increase in j as can be seen in figures 1 and 2, only detail coefficients up to roughly $j = \log_2(n)$ can be calculated, and at all levels there are discretisation effects at the edges. In analysis, as in figure 1, this results in 'polluted' coefficients (unfilled circles) whose calculation involved points in the grid for which there is no data. In generation, points such as those to the left of $k = 0$ in figure 2 have to be inserted artificially so that, beginning at $j = J$ and moving down, all coefficients necessary to generate a series beginning at $k = 0$ on level $j = 0$ are available. Note that these extra coefficients are set to zero but are necessary to avoid the recursion relations taking different forms near the edges. There are also edge effects due to truncation of $u(k)$ in the case where it is not finite.

C. A General On-Line Generation Framework

We have discussed how a sample path $x(t)$ can be recast as a set of detail coefficients in the time-scale plane, and a residual approximation sequence, and the relationships between those coefficients. Since $x(t)$ is a random process, so are its wavelet coefficients stochastic processes in their own right. Indeed since $x(t)$ is stationary each detail sequence $d_{j,\cdot}$ and approximation sequence $a_{j,\cdot}$ is a discrete time stationary process. In order to generate an accurate approximation $a_0(k) = a_{0,k}$ to $x(k)$ therefore, the statistical nature of these processes must be understood, so that samples paths from them can be generated in practice to feed the general deterministic reconstruction algorithm described above (henceforth we will treat the generation of $a_0(k)$ as our aim and ignore the final, often impossible, step to $x(t)$).

We first address the complexity of the reconstruction algorithm itself. It is easy to see that, ignoring edge effects, from a sequence $a_J(k)$ of length n_J the number of operations required to generate $a_0(k)$, which has length $n = n_J 2^J$, is $O(n_J 2^J I_u)$, where I_u is the length of the filter $u(k)$. The computational complexity is therefore linear in the length of the generated time series, which is acceptable, however

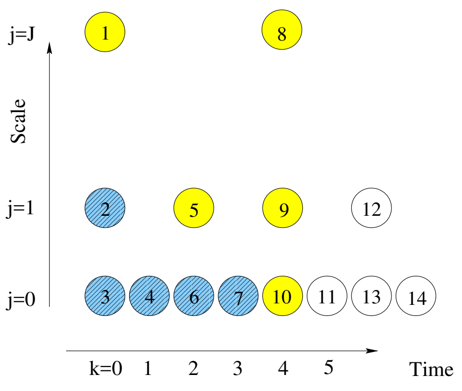


Fig. 3. **On-line generation order.** The sixth point of the output time series, $a_0(5)$, is about to be generated, the 11th coefficient overall. Of the first 10 (ignoring initialisation edge effects), those with the oblique shading are no longer needed and can be discarded, leaving only points 1, 8, 5 and 9 in memory. After point 11 is generated, 5 will be no longer needed.

so are the memory requirements of a direct off-line implementation where the a_j are calculated row by row, which is not good news for a high speed data generator. Fortunately the short range nature of the relationships between coefficients, as seen in figure 2, allow a natural on-line implementation where the coefficients are calculated in trees in *pre-order*. Figure 3 shows the order in which the first 10 coefficients are generated in a scheme where $J = 2$, and again $I_u = 4$. Note that the first tree, that bounded by points 1, 3 and 7, was generated before the second begun with 8. In this way only a small number of coefficients need ever be stored in memory, of the order of $JI_u/2 \approx (\log_2(n) - 1)I_u$. An example is detailed in the caption of the figure. Naturally, at the bottom level the emerging elements of $a_0(k)$ are the desired time series output, and are not stored.

The natural linear complexity and logarithmic memory requirements of the on-line algorithm however, although ideal for on-line analysis [23], are not sufficient for generation. For the generation as a whole to be on-line and scalable, this must be true of each of its component parts. Thus each detail process must also be amenable to an on-line implementation with no more than linear computational complexity and logarithmic memory requirements. The same is true of $a_J(k)$ at the top level, whose elements are calculated one by one as needed, although a more costly algorithm would be possible there, as the rate will be 2^J times slower than that of the output! This question of the nature and complexity of the input processes is one which is specific to the kind of final series one wishes to generate. Note however that typically the detail processes will be of the same type at each scale, though with parameters, such as variance, which will vary. The most difficult aspect of their generation however is the potential correlations across different scales. Although long term correlations are not expected – it is one of the key advantages of the wavelet representation that correlations between wavelet coefficients are weak, even when there is long memory in $x(t)$ – reproducing the correct structure both across and within scales could be time consuming. Two examples of fractal processes are given next.

D. Two Fractal Models

The first model we choose is a member of perhaps the simplest possible family with a fractal property. This property is the defining feature of second order scaling behaviour in the wavelet domain, the power-law progression of the variance of the details with j:

$$\mathbb{E}d(j,\cdot)^2 \sim 2^{j\alpha}C \tag{6}$$

where C is a positive constant. Provided $\alpha \in (0,1)$, such a process can be viewed as an approximation to the canonical LRD process, the *fractional Gaussian noise* (fGn). This approximation is extremely computationally effective, as exact generation is $O(n^2)$. The process $a_J(k)$ is set to a constant corresponding to the mean traffic rate, which here is just zero, and the details are each zero mean Gaussian IID processes, corresponding to fluctuations in the rate. That is, for each j fixed, the $d_{j,k}$ are mutually independent Gaussian random variables with zero mean and variance $2^{j\alpha}C$. Thus in this simple model there are no correlations in the wavelet plane at all, and on-line generation of the component processes is trivial. The variance of $d_{J,\cdot}$ is normalised to 1 and for $j = 1, 2, \cdots J-1$, equation (6) will be followed with $\alpha = 0.5$. Daubechies wavelets [25] with two vanishing moments, implying $I_u = 4$, are chosen, and $J = 10$. The total memory requirement of the model is around 20 floating point numbers for unlimited time series output. The program is written in `C` and only $n = 1024$ points will be generated in this example, whose aim is to illustrate the kind of processes included in the framework (see also [14], [15]). The output is shown in figure 4. In the top plot the time series $a_0(k)$ is shown, and below a *Logscale Diagram* is given, that is a wavelet based estimate of the detail variances in logarithmic coordinates, with confidence intervals [20]. In both cases the results closely resemble those obtained for actual fGn processes. Note that simply by changing α to fall within $(1,2)$, this 'IID' model family approximates another key fractal process, the *fractional Brownian motion*.

The second example follows the work of Riedi *et al*, [10], who implemented a multiplicative binomial cascade in the wavelet domain. We do not modify their approach but merely show how it fits into the on-line framework, in order to provide what is perhaps the first real-time generator of multifractal traffic, and to illustrate again the scope of the method.

Their approach is very specific in a number of ways. The first feature is that it depends fundamentally on the use of the *Haar Wavelet*. For this simplest of wavelets the scale function $\phi(t)$ is just the indicator function of the interval $[0,1]$, and the corresponding generator only has two elements: $u_0 = u_1 = 1/\sqrt{2}$, and for the wavelet $-v_1 = v_0 = 1/\sqrt{2}$; The use of Haar wavelets ensures the positivity of each approximation sequence, an essential factor in their approach, and a useful one in terms of interpreting the final series – negative traffic rates are forbidden! The second key feature is that the binomial cascade is a binomial tree structure growing from a single root. In the MRA implementation this translates in the top approximation $a_J(k)$ consisting of just a single point, a single

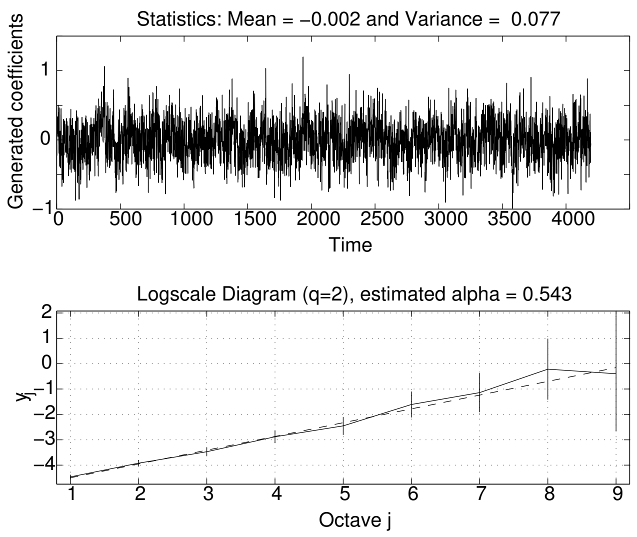


Fig. 4. **Output of an IID model** An approximation of a fGn is given by using IID Gaussian details at all scales, with variances obeying $\mathbb{E}d(j,\cdot)^2 \sim 2^{j\alpha}C$ with $\alpha = 0.5$. Daubechies wavelets with filters of length 4 were used, and $n = 1024$ points are plotted generated with $J = 10$ recursion levels. In the lower plot an estimation of the variances confirms the designed power-law.

random variable $a_J(0)$. The immediate consequence of this is that the method cannot be truly on-line, as points can only continue to flow out at level $j = 0$ if more points can be generated as needed at level J. However, by choosing J large enough, we can ensure that the finite length n of the output is as large as desired. For example, with 32 levels, and an output of one point per millisecond (assuming the machine can support this), traffic could be generated for a month, with a constant memory requirement of around 32 floating point numbers. Using the framework in this way, where we eliminate the need for an on-line algorithm for $a_J(k)$, we refer to as *Almost On Line* or 'AOL'.

The mapping of the multiplicative weights of a cascade model into the MRA structure is achieved by defining the detail processes in an unusual way, as randomly rescaled versions of their approximation coefficient ancestors:

$$d_{j,k} = R_{j,k} a_{j,k} \quad , \tag{7}$$

where the $R_{j,k}$ are symmetric random variables on $[-1, 1]$ whose distribution can vary with j but which are identically distributed for j fixed. The symmetry ensures that in an average sense, 'mass' is preserved at each stage of the construction. This method of generating the details scales well and has low computational needs. Note that because of the tree-like structure, the $R_{j,k}$ are independent along lines of descendants, but can be dependent elsewhere, unlike the complete independence of the IID example. It can be shown [10] that if the $R_{j,\cdot}$ converge in distribution as $j \to \infty$, then the output in the limit is a binomial cascade, which has known multifractal properties.

The variance decay of the details across scales can be freely controlled as follows. Let $\eta_j = \mathbb{E}[d^2_{j+1,\cdot}]/\mathbb{E}[d^2_{j,\cdot}]$ be the desired variance ratios, $0 \le j < J$. They can be calculated as

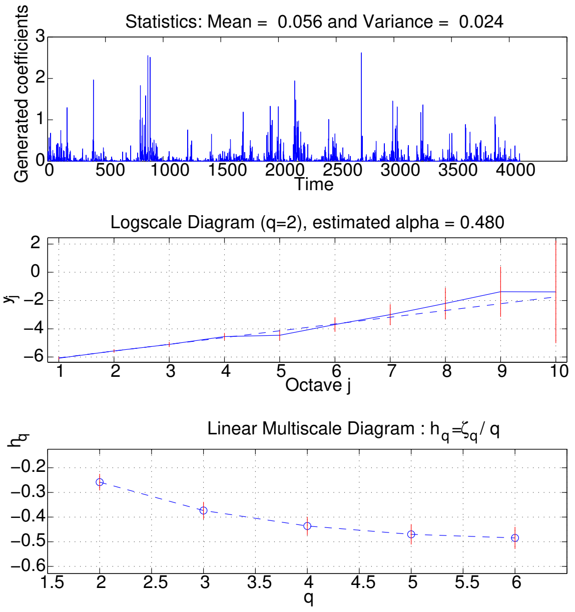


Fig. 5. **Output of a Binomial cascade model** An approximation $J = 12$ levels deep of a random conservative cascade is given using beta multipliers with shape factor $p = 10$ at the coarsest, level (nearly normal with small variance: 1/21). The detail variances decays with $\alpha = 0.5$ and $a_J(0)$ is Gaussian with mean 2 and unit variance. Top: the positive time series, Middle: estimates of the variance with scale (log coordinates), Bottom: exponent 'spectrum' plot showing non-trivial multifractal behaviour.

$$\eta_j = \frac{2\mathbb{E}[R^2_{j+1,\cdot}]}{\mathbb{E}[a^2_{j,\cdot}]\mathbb{E}[(1+\mathbb{E}[R_{j+1,\cdot})^2]} \tag{8}$$

and initialized via $\mathbb{E}[R^2_{J,\cdot}] = \mathbb{E}[d^2_{J,\cdot}/\mathbb{E}[a^2_{J,\cdot}]$. The higher order moments can also be controlled through those of the multipliers, whilst always remaining multifractal by construction.

In figure 5 an example is given of a construction $J = 12$ levels deep, with α controlled at $\alpha = 0.5$ by selecting the ratios as above. At each level the multipliers are chosen to be symmetrically beta distributed, $\beta(p(j), p(j))$, with variance $\mathbb{E}[R^2_{j,\cdot}] = 1/(2p(j) + 1)$. The value of $p(J)$ is set to give the first multiplier unit variance, and subsequently $p(j)$ evolves according to the set evolution of the detail variances, reaching a steady state in the limit and thus satisfying the above condition that the multipliers converge in distribution. Finally the variance of $a_J(0)$ is unitary and following [10] it is chosen as Gaussian (although this will cause some sample paths to be negative, their probability is low).

In the top plot the time series $a_0(k)$ is shown, which is positive and noticeably non-Gaussian. In the middle plot the *Logscale Diagram* shows the linear progession of variances with an accurate α estimate displayed in the title. The multifractal nature of the series is revealed in the lower plot, a *Linear Multiscale Diagram* [22], where scaling exponents h_q corresponding to several different moments q are plotted. These are analogous to α but have been rescaled in such a way that a horizontal alignment would indicate degenerate multifractal scaling, as would be the case for a simple fGn process. The LMD for example of the IID model is a horizontal line. The distinct curve (note the confidence intervals) shows that non-trivial multifractal scaling is indeed present as expected. Further details and other models can be found in [1].

III. From Time Series to Cell Stream

The aim of this section is to show how to take values from a time series and, by interpreting them as byte counts after an appropriate normalisation, to transmit them in real-time as a concrete traffic stream whose measured bit rate matches that of the time series as closely as possible.

For practical reasons of accessiblity we concentrate on ATM over SONET (Synchronous Optical NETwork), a common networking solution. However, we do not wish to constrain our methods to one specific technology. Hence we layer the approach by first translating the time series into a sequence of TCP/IP packets for subsequent passing to the transmission layer. In this way the lower layer could be changed without affecting much of the system. Given the looming ubiquity of the TCP/IP suite we expect this to be a useful approach, especially as traffic models will increasingly aim to model TCP/IP traffic directly.

A. Background

We must first understand the specifics of the network we are using. We consider a TCP/IP over ATM over SONET network. That is, TCP packets encapsulated in IP packets are transmitted over the ATM Adaptation Layer 5 (AAL5) as ATM cells, which are then sent across OC3 SONET frames at a nominal 155 Mbps. Typically, each TCP/IP packet is packaged as an AAL5 Protocol Data Unit (PDU) by prepending a short LLC/SNAP header which identifies that TCP/IP is being carried, and appending a short trailer. The PDU is then segmented into 48 byte chunks and a 5 byte header attached to each to form 53 byte ATM cells, as illustrated in figure 6. A typical TCP/IP packet has a 40 byte header, and the LLC/SNAP header is 8 bytes, so all the header information is held in the first ATM cell, whilst the trailer information is held in the last 8 bytes of the final cell. A bit in the ATM header identifies it as the last cell of the PDU.

Our desire to build an inexpensive, simple traffic generator strongly suggested that the monitor be based on a PC architecture, using a commodity ATM Network Interface Card (NIC). We used a 330 Mhz machine running `FreeBSD`. The FORE Systems PCA-200EPC ATM NIC has been used for network monitoring and traffic generation by MCI on the vBNS (very high performance Backbone Network Service [28], [29]). In fact the CORAL group [5], [6], [16], [18], specifically the OC3MON project [19] have made the drivers for this NIC freely available, and they form the basis for our work. Specifically, the existing `DOS` transmit driver and the `FreeBSD` receive driver (the basis of our traffic measurement infrastructure [30]) served as templates for a new `FreeBSD` transmit driver, a task seriously complicated by the fact that details of the cards' workings and source for its firmware were unavailable. The resulting `C` code was added to the kernal source file for the existing

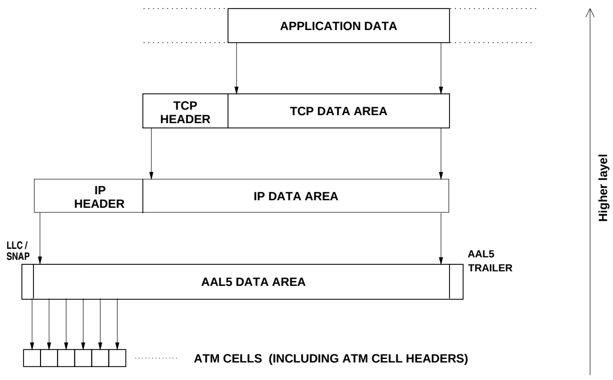


Fig. 6. **Protocol Encapsulation.** Application data is first packetised into TCP packets transported by IP packets. For transmission these are assembled into AAL5 data units and finally split into ATM cells, ready to be physically sent via SONET.

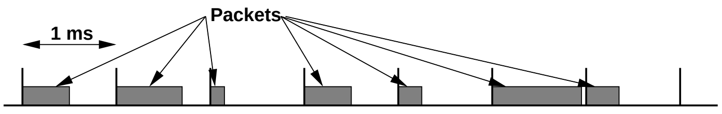


Fig. 7. **Translation of time series to packet stream.** A single packet is sent at the beginning of each transmission interval (one per time series element), which is 348 cells ≈ 1ms long.

`FreeBSD` receive driver and can be used by the `FreeBSD` CORAL code. Some bugs were also fixed, and of course some details are specific to our application (see [2]).

The use of such hardware allowed us to build a very powerful traffic generator for around $4000 Australian. In principle, multiple NICs could be used, allowing multiple traffic generators (and monitors) to run within the same PC, further reducing costs.

B. From time series to packet stream

We have chosen a very simple, yet effective approach for turning our time series into a packet stream. The basic idea, illustrated in figure 7, is to transmit single packets at uniformly separated *transmission intervals*, each corresponding to an element of the time series. The variable bit/cell rate is achieved by changing the packet sizes. This has large processing advantages described in detail below.

The target transmission interval was chosen to be 1 ms. In fact cells are the natural time unit here as they dictate the finest resolution available for the final transmitted data rate. One transmission interval was therefore set to 348 cells, which is just under 1 ms, and packets were generated according to this quantisation. There are therefore 349 possible packet sizes and corresponding rate values (including zero). If the transmission interval were smaller, which is desirable from the point of view of specifying the traffic rate very precisely, this quantisation would become noticeable and eventually all details of the time series would be lost. Furthermore when the sampling interval is smaller the processing load on the system is increased. It was found that 1ms gave an acceptable quantisation error, and good performance.

The steps in creating such a packet stream are:

1. **Normalize time series to desired traffic rate.** The

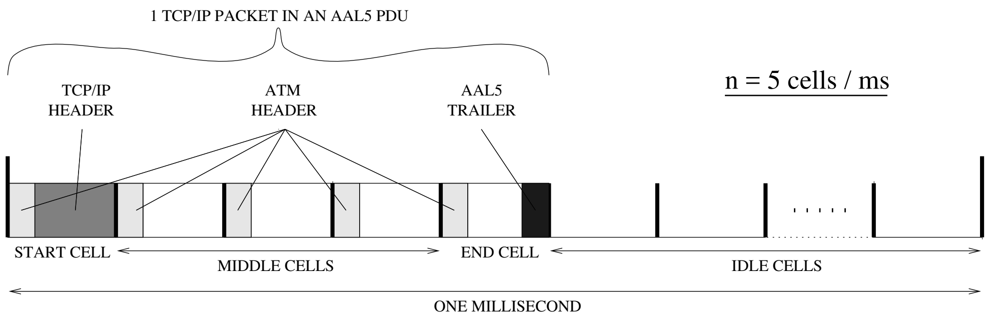


Fig. 8. **Structure of the 348 cell long transmission interval** A single time series value is translated to a single TCP packet of the corresponding size, to be sent in one transmission interval, $\approx$ 1ms. The packet is converted to a sequence of 348 modified ATM cells forming a pseudo AAL5 PDU.

unitless time series may be arbitrarily normalized. Either by selecting the input parameters to the time series generator, and/or by linearly transforming the output series using user input parameters such as the mean and variance, a normalized packet size sequence should be outputted in units of cells per transmission interval.

2. **Map the packet rates into the quantisation.** The packet size sequence is still real valued and must be mapped to the discrete set $\{0, 1, \cdots 348\}$. Values are first rounded up to the nearest integer, then negative values set to zero (many traffic models allow unphysical negative rates), then excess 'cells' over 348 are virtually buffered, that is, they are stored until the first interval with spare capacity. In this way the final traffic will reflect the real effects of traffic bursts exceeding the output bandwidth (see figure 9).
3. **Create TCP/IP packet headers.** A template packet header specified at the start of the program, containing the IP addresses, TCP ports etc. is copied to each TCP/IP packet header. The packet length is calculated from the known total length of the packet in cell units and inserted in the header, and finally the CRC is calculated.
4. **Assemble the TCP/IP packets.** This is not done in our implementation as we assemble the AAL5 PDU directly, as described below. In any case for our purposes the packets do not carry real data, they will be, in effect, stuffed with zeros.

The program which performs these steps can be called from the unix command line and acts as a filter; that is, it reads the time series form standard input. It then generates the packet sizes, and headers which are passed to a lower level process to create and send the packets. This approach is very flexible, it allows time series to be generated by another program, or read from a file. Hence a particular time series can be used again and again, or a very long time series can be generated on-line continuously as we require.

C. From packet stream to cell stream

The previous process sent a series of requests to transmit packets, which now need to be sent to the transmission layer, in this case ATM. However, it would be time consuming and inefficient to try to transmit each of these packets individually. It would also be very hard to guarantee that the packets are transmitted exactly 1 transmission interval apart.

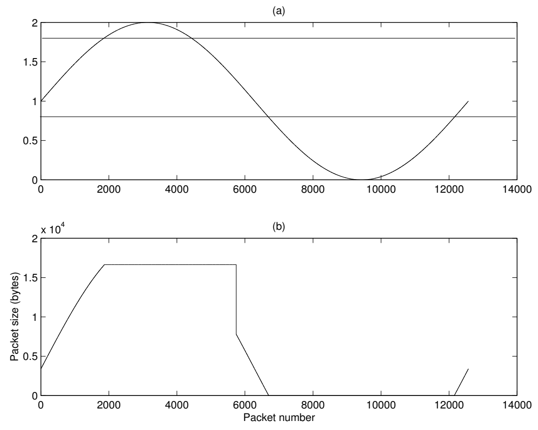


Fig. 9. **Effect on rate of quantisation and virtual buffering.** A time series defining the un-normalised traffic rate (top), is converted into a packet stream, one per $\approx$ 1ms transmission interval (bottom). After normalisation the zero (resp. maximum) traffic rate is set to the central (resp. upper) horizontal line in the top plot. The resulting quantisation effect (truncation below the zero line) and virtual buffering (excess above the upper line being carried over) are visible in the packet stream. Within the two lines however the 349 rate levels gives good resolution.

The CORAL project provided the basis for a much more efficient approach. The FORE card integrates the AAL5, ATM and SONET layers, however rather than passing packets to it we took advantage of the fact that CORAL have rewritten the firmware to allow the transmission of a precisely defined cell stream. Sets of pseudo AAL5 PDU's can be constructed externally and placed in a buffer. The ATM card is programmed to read from the buffer using Direct Memory Access (DMA) and transmit them continuously, as long as buffer 'blocks' are available. The 'cells' in these pseudo PDU's differ from normal ATM cells. First the 1 byte HEC checksum is absent as it is later added by the card and second, a four byte `count` field is prependend which tells the card how many times to send that particular cell. These constructed 'cells' are thus $53 - 1 + 4 = 56$ bytes long. The `count` field means that the amount of data which need be written to the buffer is greatly reduced, and furthermore, that the load on the packet generator process above, and the PDU creation process, as well as the reading time per PDU, is essentially independent of traffic rate but depends only on the chosen transmission interval, an inherent scaling feature.

Figure 8 illustrates the construction of one pseudo PDU, corresponding to the transmission of a single TCP/IP packet 5 cells in length over a transmission interval 348 cells long. It consists of a header cell containing the TCP/IP header described above together with the LLC/SNAP header, a trailer cell containing the AAL5 trailer at the end, and 3 empty body cells inbetween. Through the `count` field, only one body cell need actually be written, with a `count` value of 3. Similarly, the special blank cell, which tells the card to send nothing for a cell-period, need only be written once, with a `count` value of $348 - 5 = 343$. It is possible to send no packet in one transmission interval (blank cell with a count of 348), and a packet just one cell long (a special cell can be written including both the TCP/IP header and the AAL5 trailer, with a count of 1).

The idea is that continuous reading of the pseudo PDU's by the card will result in a totally controlled output cell stream, with no gaps. Good buffer management is clearly important to ensure this. Because of the use of DMA at least two buffer blocks are needed to safely separate reading and writing. In fact 5 were chosen each 9325 cells long. These were sufficiently large that request interupts, generated when passing from one block to another, were not generated too often, and 5 seems sufficient to smooth out scheduling effects of the user processes without taking up too much memory. The management itself has some subtleties (see [2] for details) but essentially a series of pseudo PDU's are written to an available buffer block until full. It is then marked as full and writing immediately moves to the next free block. When the NIC finishes reading a buffer it interrupts to find the address of the next full block and moves to it. The NIC sends from an already loaded block as it reads the next, so that no gaps form.

IV. The Working System

A. A Multifractal Example

The two independent components of the system, the on-line time series generator and the packet-sender and kernal routines, are linked simply via unix pipes. The output of the card is sent over an optical link to another OC3Mon based measurement system, with another FORE Systems card, as described in [30]. This system operates on a 'first cell of TCP/IP packet' mode, where only 1 cell per packet is measured, and approximate bits rates are obtained with the help of the `Total Length` field in the header. This does not result in any loss of accuracy here, given the known structure of the cell stream.

An example of the output, using the multifractal model described in the previous section, is given in figure 10. In contrast to figure 9, where the normalisation was chosen to result in extreme quantisation and normalisation effects, here the normalisation was chosen to fit the quantisation well. The measured traffic, at least to the eye, appears to be an exact copy in cells per millisecond of the original target time series generated at the sender. Note that the vertical scale is in cells per transmission interval, and that high rates were achieved, in one case a burst comes very close to the maximum of 348.

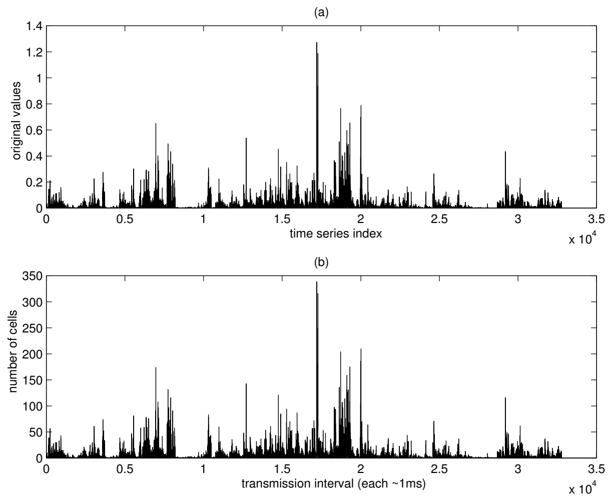


Fig. 10. **Multifractal time series and received cell rate.** The positive, normalised time series (top) is converted to a cell stream, sent over an optic link, measured via first cell of TCP/IP packet measurement mode, and plotted at the finest resolution possible, one point per transmission interval. We see that the reproduction is excellent.

Testing has shown that all cells sent to the card were received at the receiver. Jitter in the packet arrival times was measured with a GN Nettest Interwatch 95000 traffic monitor, and was found to be no greater than $\pm 5\mu$s.

B. Scalability

As mentioned above, the packet and PDU generation processes, which are user processes, have workloads which are independent of traffic rate. Furthermore their workloads are small, for example only four pseudo cells (and sometimes only 3, 2 or even 1) need to be written to a buffer block per millisecond. Furthermore it was found that very heavy workloads were needed before 5 buffer blocks were insufficient to smooth out the effects of other processes on the 330 Mhz machine used. The workload of the kernal is negligeable, being restricted to the notification of full blocks, as DMA is used. Problems therefore can only arise if the transmission interval is made too small. Decreasing this interval increases the rate at which the card must read the blocks, and therefore the kernal interrupts and the writing rate of the pseudo PDU's in direct proportion. A transmission interval of 1ms was very easily supported by our modest PC. Higher rates were not tried as the quantisation would be too coarse at OC3 rates. We believe that the same system could support OC12 rates with a transmission interval of at least as small as 100μs, assuming that the on-line time series generator could deliver values at this rate also. This would depend on the speed of the machine, the number of competing processes, and the computational details of the detail process as described in the last section.

V. Conclusion

We have successfully combined two independent systems to send traffic according to a sophisticated multi-fractal traffic model across an ATM OC3 link in real time. Rates close to link saturation are possible for indefinite periods, as the load on the sending processes are independent of rate, they are functions only of the rate resolution, the 'transmission interval'. This resolution can be freely chosen, and was set here to 348 cells, which is just under 1ms, and no processing difficulties we encountered. This rate is a reasonable compromise between a fine resolution specification of traffic rate, and quantisation effects of the target rate values from the time series due to the size of ATM cells.

The first system is an on-line wavelet synthesis framework, implemented in *C*. Many different traffic models can be incorporated in it, and it is particularly suited to the generation of fractal models. Examples were given of an approximate fractional Gaussian noise, a long-range dependent process, and an exact implementation of a simple multifractal model proposed in [10] (figure 10). Data can be generated indefinitely with trivial memory requirements, and could be used for a variety of purposes, for example to drive simulations. The only difficulty, a topic for continued research, is the specification and on-line generation of the appropriate *detail processes*, which depend on the desired traffic model.

The second system takes an input time series in real time, normalises it, and converts it to an equivalent TCP/IP packet stream at a rate of one packet at the beginning of each transmission interval (this could be easily altered to spread the traffic out more evenly). It then directly constructs an AAL5 data unit in memory which carries the packet. The firmware on the FORE Systems ATM network interface card used, which the CORAL project has made freely available, reads these PDU's from a buffer, and transmits them. A new `FreeBSD` driver was written to make use of this firmware in transmit mode. Through sensible buffering it was possible to send a continuous supply of PDU's to the card, without gaps. Furthermore the entire system has a workload which is independent of the traffic rate, thanks to the ability to give the card an instruction to send the same cell multiple times. The system is therefore inherently scalable.

References

[1] J. Bäkar, "A Framework for implementing fractal traffic models in real-time" Master's thesis, SERC Melbourne, 1999.

[2] J. Wall, "Implementing a tool for converting time series into ATM traffic" Master's thesis, SERC Melbourne, 1999.

[3] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 1–15, Feb 1994.

[4] V. Paxson and S. Floyd, "Wide-area traffic: The failure of poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, 1994, `http://ee.lbl.gov/nrg-papers.html`.

[5] "Coral network traffic analysis," online: `http://moat.nlanr.net/Coral/`.

[6] J. Dugan, "Coral – flexible, affordable, high performance network statistics collection," online: `http://www.caida.org/Tools/Coral/`.

[7] A. Feldmann, A.C. Gilbert, W. Willinger, Data networks as cascades: Investigating the multifractal nature of Internet WAN traffic. Proceedings of the ACM/SIGCOMM'98, Sept. 1998, Vancouver, Canada.

[8] J. Beran, R. Sherman, M. S. Taqqu, and W. Willinger, Long-range dependence in variable-bit-rate video traffic, *IEEE Trans. on Comm.*, 43:1566–1579, 1995.

[9] R.H. Riedi, An improved multifractal formalism and self-similar measures, *J. Math. Anal. Appl., Vol.189*, pp.462–490, 1995.

[10] R. Riedi, M.S. Crouse, V.J. Ribeiro, R.G. Baraniuk, A Multifractal Wavelet Model with Application to Network Traffic, *IEEE Trans. Inf. Th. special issue on "Multiscale Statistical Signal Analysis and its Applications"*, vol. 45, no. 3, April 1999.

[11] R. Riedi, J.L. Lévy Véhel, TCP traffic is multifractal: a numerical study, Preprint 1999.

[12] Bong K. Ryu and Mahesan Nandikesan, "Real-time generation of fractal atm traffic: Model, algorithm and implementation," Tech. Rep. 440-96-06, Center for Telecommunications Research, Columbia University, New York, 1996, Available at `http://comet.columbia.edu/comet/research/control/traffgen.html`.

[13] A. A. Lazar and M. Nandikesan, "Real-time traffic generation and QOS monitoring system," Tech. Rep. 481-97-15, Center for Telecommunications Research, Columbia University, New York, 1997, Available at `http://comet.columbia.edu/comet/research/control/traffgen.html`.

[14] Sheng Ma and Chuanyi Ji, "Modeling video traffic using wavelets," in *ICC'98*, 1998.

[15] Sheng Ma and Chuanyi Ji, "Modeling video traffic in the wavelet domain," in *IEEE Infocom'98*, 1998.

[16] G.J. Miller, K. Thompson, and R. Wilder, "Performance measurement on the vBNS," in *Proceedings of the Interop'98 Engineering Conference*, Las Vegas, NV, May 1998.

[17] D. Veitch and M. S. Taqqu and P. Abry, "Meaningful MRA initialisation for discrete time series" To appear, *Signal Processing*, 2000.

[18] Kevin Thompson, Gregory J. Miller, and Rick Wilder, "Wide-area Internet traffic patterns and characteristics," *IEEE Networks*, 1997, Extended Version: `http://www.vbns.net/presentations/papers/index.html`.

[19] J. Apisdorf, K. Claffy, K. Thompson, and R. Wilder, "OC3MON: Flexible, affordable, high performance statistics collection," in *INET'97 Conference*, June 1997.

[20] Darryl Veitch and Patrice Abry, "A wavelet based joint estimator of the parameters of long-range dependence," *IEEE Trans. Inf. Th. special issue on "Multiscale Statistical Signal Analysis and its Applications"*, vol. 45, no. 3, April 1999.

[21] P. Abry and D. Veitch, "Wavelet analysis of long-range dependent traffic," *IEEE Trans. on Info. Theory*, vol. 44, no. 1, pp. 2–15, 1998.

[22] P. Abry, P. Flandrin, M.S. Taqqu and D. Veitch. Wavelets for the analysis, estimation and synthesis of scaling data. To appear in *Self-Similar Network Traffic and Performance Evaluation*, K. Park and W. Willinger, eds., Wiley Interscience, 1999.

[23] Matthew Roughan, Darryl Veitch, and Patrice Abry, "On-line estimation of parameters of long-range dependence," in *IEEE GLOBECOM'98*, Sydney, Australia, Nov. 1998, pp. 3716–3721.

[24] P. Abry, P. Gonçalvès, and P. Flandrin, *Wavelets and Statistics*, vol. 105 of *Lecture Notes in Statistics*, chapter Wavelets, Spectrum estimation, $1/f$ processes., pp. 15–30, Springer-Verlag, New York, 1995.

[25] I. Daubechies, *Ten Lectures on Wavelets*, SIAM, Philadelphia (PA), 1992.

[26] A.Feldmann, A.C.Gilbert, W.Willinger, and T.G.Kurtz, "The changing nature of network traffic: Scaling phenomena," *Computer Communications Review*, vol. 28, no. 2, 1998.

[27] D. Veitch, P. Abry, P. Flandrin and P. Chainais. "Infinitely divisible cascade analysis of network traffic data," to appear, proceedings of ICASSP 2000, Istanbul Turkey, June 2000.

[28] "Very high performance backbone network service," `http://www.vbns.net/`.

[29] "The university of waikato, atm group page," online: `http://atm.cs.waikato.ac.nz/atm/`.

[30] Matthew Roughan, Darryl Veitch, Martin Ahsberg, Hans Elgelid, Maurice Castro, Mick Dwyer, Patrice Abry "Real-Time Measurement of Long-Range Dependence in ATM Networks" in *PAM2000, Workshop on Passive and Active Networking, New Zealand*, 2000.

[31] Bong K. Ryu and Steven B. Lowen, "Point process approaches to the modelling and analysis of self-similar traffic - part i: Model construction," in *IEEE INFOCOM'96: The Conference on Computer Communications*, San Francisco, California, March 1996, vol. 3, pp. 1468–1475, IEEE Computer Society Press, Los Alamitos, California.

[32] M.Roughan, J.Yates, and D.Veitch, "The mystery of the missing scales: Pitfalls in the use of fractal renewal processes to simulate lrd processes," in *Applications of Heavy Tailed Distributions in Economics, Engineering and Statistics*, American University, Washington, DC, June 1999.

[33] Steven B. Lowen and Malvin C. Teich, “Estimation and simulation of fractal stochastic point processes,” *Fractals*, vol. 3, no. 1, pp. 183–210, 1995.

[34] Matthew Roughan and Darryl Veitch, “A study of the daily variation in the self-similarity of real data traffic,” in *Proceedings of the 16th International Teletraffic Congress - ITC 16*, P. Key and D. Smith, Eds. 1999, vol. 3b, pp. 67–76, Elsevier, Amsterdam.

[35] P. Abry, F. Sellan, “The wavelet based synthesis for fractional Brownian motion proposed by F. Sellan and Y. Meyer: remarks and implementation,” in *Applied and Computational Harmonic Analysis*, 3:377–383, 1996.